2153

TFW

LAW OFFICES
DOWELL & DOWELL, P.C.
PATENT AND TRADEMARK CAUSES
SUITE 406
2111 EISENHOWER AVE.
ALEXANDRIA, VA 22314

A. YATES DOWELL, III
RALPH A. DOWELL

WENDY M. SLADE
REGISTERED PATENT AGENT

TELEPHONE (703) 415-2555
FACSIMILE (703) 415-2559
E-MAIL: dowell@dowellpc.com

February 13, 2006

US Patent and Trademark Office
P O Box 1450
Alexandria, VA 22313-1450

Re: Office Action for SN 09/941,619
Comeau

Gentlemen:

We are returning the Office Action issued on February 10, 2006 with respect to the above referenced application. We had requested and received a granted petition to be removed as attorney-of-record. A copy of the granted petition is enclosed.

All future correspondence was to be sent to ZUCOTTO WIRELESS INC. as noted on the second page of the granted petition.

Very sincerely yours

Ralph A. Dowell

RAD:pgs
Enclosures Office Action

UNITED STATES PATENT AND TRADEMARK OFFICE

COMMISSIONER FOR PATENTS
UNITED STATES PATENT AND TRADEMARK OFFICE
P.O. BOX 1450
ALEXANDRIA, VA 22313-1450
www.uspto.gov

Paper No. 5

R. Allan Brett
DOWELL & DOWELL PC
Suite 309
1215 Jefferson Davis Highway
Arlington, VA 22202

In re Application of:
Guillaume Comeau, et al.
Application No. 09/941,619
Filed: August 30, 2001
For: METHOD AND APPARATUS FOR INTERPROCESSOR COMMUNICATION AND PERIPHERAL SHARING

DECISION ON REQUEST TO WITHDRAW AS ATTORNEY OR AGENT

This is a decision on the Request To Withdraw from Representation filed March 17, 2004.

A grantable request to withdraw as attorney of record should indicate thereon the present mailing addresses of the attorney(s) who is/are withdrawing from the record and of the applicant. The request for withdrawal must be signed by every attorney seeking to withdraw or contain a clear indication that one attorney is signing on behalf of another/others. A request to withdraw will not be approved unless at least 30 (thirty) days would remain between the date of approval and the later of the expiration date of a time to file a response or the expiration date of the maximum time period which can be extended under 37 C.F.R. § 1. 136(a). The effective date of withdrawal being the date of decision and not the date of request. See M.P.E.P. § 402.06. 37 C.F.R. § 1.36 further requires that the applicant or patent owner be notified of the withdrawal of the attorney or agent.

The request filed March 17, 2004 meets all the requirements. Accordingly the request is **GRANTED**.

All future communications from the Office will be directed to the below-listed address until otherwise notified by applicant. This correspondence address is provided by the withdrawn attorney(s). Applicant is reminded of the obligation to promptly notify the Patent and Trademark Office (Office) of any change in correspondence address to ensure receipt of all communications from the Office.

James R. Matthews

Vincent N. Trans
Special Program Examiner
Technology Center 2100
Computer Architecture, Software, and
Information Security
703-305-9750

cc: ZUCOTTO WIRELESS INC.
16644 West Bernardo Drive
Suite 301
San Diego, CA 92127

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/941,619	08/30/2001	Guillaume Comeau	78524-9 /jlo	6743

293 7590 02/10/2006

Ralph A. Dowell of DOWELL & DOWELL P.C.
2111 Eisenhower Ave
Suite 406
Alexandria, VA 22314

EXAMINER	
NASH, LASHANYA RENEE	
ART UNIT	PAPER NUMBER
2153	

DATE MAILED: 02/10/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

MAIL
FEB 1 7 2006
RECEIVED

Office Action Summary	Application No.	Applicant(s)	
	09/941,619	COMEAU ET AL.	
	Examiner	Art Unit	
	LaShanya R. Nash	2153	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *30 August 2001*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-86* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-86* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

Claims 1-86 are being considered.

Claim Objections

Claims 2-7 are objected to because of the following informalities: inconsistent terminology, and grammatical errors. Appropriate correction is required.

Examiner suggests replacing "The resource sharing system" recited in line 1 of claims 2-6 with "The system", so as to maintain consistent terminology with subsequent dependent claims.

Examiner suggests replacing "second processor core", recited in line 6 of claim 7 with "second processor", so as to maintain consistent terminology.

Examiner suggest replacing "A system", recited in line 1 of claims 30-31 with "The system", so as to maintain consistent terminology with subsequent dependent claims.

Examiner suggests replacing "first processing means" recited in line 2 of claim 80 with "a first processing means", so as to maintain consistent terminology.

Examiner suggests inserting "and" after semi-colon in line 4 of claim 84.

Claim Rejections - 35 USC § 102

The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

Claims 60-75 is rejected under 35 U.S.C. 102(b) as being anticipated by Chennubhotla et al. (US Patent 5,841,988), hereinafter referred to as Chennubhotla.

In reference to claim 60, Chennubhotla discloses an interprocessor communication interface to facilitate the expedient and efficient exchange of information between processing elements in a multiprocessing environment, (abstract; column 1, lines 42 to column 2, line 6). Chennubhotla explicitly discloses:

- An interprocessor interface (Figures 1&2-item 2; columns 2-3) for interfacing between a first processor core (i.e. processing subsystem; Figure 1-item 12-1; columns 4-5) and a second processor core (i.e. processing subsystem; Figure 1-item 12-2), the interprocessor interface comprising:
- At least one data FIFO queue (i.e. receive FIFO buffer; Figure 2-item 126) having an input adapted to receive data from the second processor core and an output adapted to send data to the first processor core;
- At least one data FIFO queue (i.e. transmit FIFO buffer; Figure 2-item 124) having an input adapted to receive data from the first processor core and an output adapted to send data to the second processor core;
- A first out-of-band message transfer channel (i.e. interprocessor communications bus; Figures 1&2-item 18) for sending a message from the first processor core to the second processor core;
- A second out-of-band message transfer channel (i.e. local bus; Figure 2-item 108) for sending a message from the second processor core to the first processor core, (columns 4-6).

In reference to claim 61, Chennubhotla shows on a chip comprising an interprocessor interface in combination with the second processor core, (columns 1-5).

In reference to claim 62, Chennubhotla shows an interprocessor interface according to further comprising: a first interrupt channel adapted to allow the first processor core to interrupt the second processor core; and a second interrupt channel adapted to allow the second processor core to interrupt the first processor core, (columns 6-10; Figure 2-item 128).

In reference to claim 63, Chennubhotla shows an interprocessor interface according further comprising at least one register adapted to store an interrupt vector, (Figure 2-item 122; columns 6-10).

In reference to claim 64, Chennubhotla shows an interprocessor interface having functionality accessible by the first processor core memory mapped to a first memory space understood by the first processor core, and having functionality accessible by the second processor core memory mapped to a second memory space understood by the second processor core, (columns 5-6; Figure 2-item 20)

In reference to claim 65, Chennubhotla shows an interprocessor interface comprising a first access port comprising: a data port, an address port and a plurality of control ports, (columns 5-6; Figure 2-item 20)

In reference to claim 66, Chennubhotla shows an interprocessor interface wherein the control ports one or more of a group comprising chip select, write, read, interrupt, and DMA (direct memory access) interrupts, (columns 5-6; Figure 2-item 20).

In reference to claim 67, Chennubhotla shows an interprocessor interface further comprising chip select decode circuitry adapted to allow a chip select normally reserved for another chip to be used for the interprocessor interface over a range of addresses memory mapped to the interprocessor interface the range of addresses comprising at least a sub-set of addresses previously mapped to said another chip, (columns 6-9).

In reference to claim 68, Chennubhotla shows an interprocessor interface comprising a second access port comprising: a data port, an address port, and a control port, (columns 5-6; Figure 2-item 20).

In reference to claim 69, Chennubhotla shows an interprocessor interface in combination with the second processor, wherein the second access port is internal to the system on a chip, (columns 5-6; Figure 2-item 20).

In reference to claim 70, Chennubhotla shows an interprocessor interface further comprising at least one general-purpose input/output pin, (columns 5-10; Figure 2-item 122).

In reference to claim 71, Chennubhotla shows an interprocessor interface further comprising: a first plurality of memory mapped registers accessible to the first processor core, and a second plurality of memory mapped registers accessible to the second processor core, (columns 5-10; Figure 2-item 20).

In reference to claim 72, Chennubhotla shows an interprocessor interface wherein the second processor core has a sleep state in which the second processor core has a reduced power consumption, and in which the interprocessor interface remains active, (columns 7-12).

In reference to claim 73, Chennubhotla shows an interprocessor interface further comprising a register indicating the sleep state of the second processor core, (columns 5-7; Figure 2).

In reference to claim 74, Chennubhotla shows an interprocessor interface wherein the second processor core has a sleep mode in which the second processor core has a reduced power consumption, and in which the interprocessor interface remains active, (columns 7-12).

In reference to claim 75, Chennubhotla shows an interprocessor interface further comprising a register indicating the sleep state of the second processor core, (columns 7-12).

Claim Rejections - 35 USC § 103

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

Claims 1-7,11-20, 27-29,50-59, and 80-82 rejected under 35 U.S.C. 103(a) as being unpatentable over Chennubhotla above, and further in view of Dutta (US Patent Application Publication 2001/0047383), hereinafter referred to as Chennubhotla and Dutta.

In reference to claim 1, Chennubhotla discloses a system for interprocessor communication, in order to facilitate the expedient and efficient exchange of information between processing elements in a multiprocessing environment, (abstract; column 1, lines 42 to column 2, line 6). Chennubhotla explicitly discloses:

- A resource sharing system (i.e. multiprocessing system sharing system memory; Figures 1& 2;columns 2-3) comprising:
- A first processor (Figure 1-item 12-1) and a second processor (Figure 1-item 12-2), the first processor managing a resource (i.e. system memory; Figures 1&2-item 14) which is to be made available to the second processor (i.e. each processing subsystem processes instruction code out of system memory; column 4);
- A physical layer interconnection between the first processor and the second processor (i.e. interprocessor communications bus; Figures 1&2-item 18) a first application layer entity (i.e. data processing function/DSP Engine; Figure 2-items 102&103; column 1; column 5, lines 16-32) on the first processor and a corresponding second application layer entity on the second processor (i.e. DSP Engine; Figure 2-items 102&103), the first application layer entity and the second application layer entity together being adapted to arbitrate access (i.e. via the Memory controller/DMA; Figure 2-item 104) to the resource between the first processor and the second processor using the interprocessor communications protocol, (i.e. interprocessor communications protocol; columns 1-3; column 7), the physical layer interconnection and the intercommunications protocol to provide a communication channel between the first application

layer entity and the second application layer entity, (i.e. interprocessor communications interface and protocol used to pass data between processing elements; columns 4-6).

However, Chennubhotla fails to disclose the system comprises: a communications protocol comprising a first interprocessor communications protocol running on the first processor, and a second interprocessor communications protocol running on the second processor which is a peer to the first interprocessor communications protocol; and arbitrating access to the resource between the first processor and the second processor using the first interprocessor communications protocol, and the second intercommunications protocol to provide a communication channel between the first application layer entity and the second application layer entity. However, it would have been obvious to accordingly modify the aforementioned system, as disclosed by Chennubhotla, for one of ordinary skill in the art at the time of the invention, as further evidenced by Dutta.

In an analogous art, Dutta discloses a system for interprocessor communications between a processor and legacy devices operating under distinct protocols, (abstract). Dutta further discloses that the system comprises: a communications protocol (i.e. translation protocol; paragraphs [0030]-[0033]), comprising a first interprocessor communications protocol (i.e. common communications protocol; paragraph [0028]) running on the first processor (i.e. client computer; Figure 3-item 21; Figure 6), and a second interprocessor communications protocol (i.e. proprietary protocols; paragraph [0029]) running on the second processor (i.e. legacy device; Figure 2-item 17; Figure 3-item 47; Figure 8) which is a peer to the first interprocessor communications protocol; and arbitrating access using the first interprocessor communications protocol, and the second intercommunications protocol to provide a communication channel between the first application layer entity (i.e. application programs; paragraphs [0050]-[0054]; Figure 6-item 124) and the second application layer entity (i.e. legacy programs; paragraphs [0062]-[0066]; Figure 8-item 224), (paragraphs [0023]-[0033]). Thusly, one of ordinary skill in the art would have been motivated to accordingly modify the aforementioned system as disclosed by Chennubhotla, so as to support communicate interprocessor communication with legacy devices and thereby increasing system compatibility with various legacy devices available in the distributed environment, (Dutta; paragraphs [0003]-[0007]).

In reference to claim 80, Chennubhotla discloses a system for interprocessor communication, in order to facilitate the expedient and efficient exchange of information between processing elements in a multiprocessing environment, (abstract; column 1, lines 42 to column 2, line 6). Chennubhotla explicitly discloses:

- A resource sharing system (i.e. multiprocessing system sharing system memory; Figures 1& 2;columns 2-3) comprising:
- A first processing means (i.e. processing subsystem; Figure 1-item 12-1) and a second processing means (i.e. processing subsystem; Figure 1-item 12-2), the first processing means managing a resource (i.e. system memory; Figures 1&2-item 14) which is to be made available to the second processing means (i.e. each processing subsystem processes instruction code out of system memory; column 4);
- An interprocessor communications protocol means (i.e. interprocessor communications interface and protocol; Figure 1-item 20; columns 2-3);
- A physical layer interconnection means between the first processing means and the second processing means (i.e. interprocessor communications bus; Figures 1&2-item 18) a first application layer means (i.e. data processing function/DSP Engine; Figure 2-items 102&103; column 1; column 5, lines 16-32) on the first processing means and a corresponding second application layer means on the second processing means (i.e. DSP Engine; Figure 2-items 102&103), the first application layer means and the second application layer means together being adapted to arbitrate access (i.e. via the Memory controller/DMA; Figure 2-item 104) to the resource means between the first processing means and the second processing means using the interprocessor communications protocol means, (i.e. interprocessor communications protocol; columns 1-3; column 7), the physical layer interconnection means and the intercommunications protocol means to provide a communication channel between the first application layer means and the second application layer means, (i.e. interprocessor

communications interface and protocol used to pass data between processing elements; columns 4-6).

However, Chennubhotla fails to disclose the system comprises: a first interprocessor communications protocol means running on the first processing means, and a second interprocessor communications protocol means running on the second processing means which is a peer to the first interprocessor communications protocol means; and arbitrating access to the resource between the first processing means and the second processing means using the first interprocessor communications protocol means, and the second intercommunications protocol means to provide a communication channel between the first application layer means and the second application layer means. However, it would have been obvious to accordingly modify the aforementioned system, as disclosed by Chennubhotla, for one of ordinary skill in the art at the time of the invention, as further evidenced by Dutta.

In an analogous art, Dutta discloses a system for interprocessor communications between a processor and legacy devices operating under distinct protocols, (abstract). Dutta further discloses that the system comprises: a communications protocol means (i.e. protocol translator; paragraphs [0030]-[0033]), comprising a first interprocessor communications protocol means (i.e. common communications protocol; paragraph [0028]) running on the first processing means (i.e. client computer; Figure 3-item 21; Figure 6), and a second interprocessor communications protocol means (i.e. proprietary protocols; paragraph [0029]) running on the second processing means (i.e. legacy device; Figure 2-item 17; Figure 3-item 47; Figure 8) which is a peer to the first interprocessor communications protocol means; and arbitrating access using the first interprocessor communications protocol means, and the second intercommunications protocol means to provide a communication channel between the first application layer means (i.e. application programs; paragraphs [0050]-[0054]; Figure 6-item 124) and the second application layer means (i.e. legacy programs; paragraphs [0062]-[0066]; Figure 8-item 224), (paragraphs [0023]-[0033]). Thusly, one of ordinary skill in the art would have been motivated to accordingly modify the aforementioned system as disclosed by Chennubhotla, so as to support communicate interprocessor communication with legacy devices and thereby increasing system compatibility with various legacy devices available in the distributed environment, (Dutta; paragraphs [0003]-[0007]).

In reference to claim 2, Dutta shows the resource sharing system wherein arbitrating access to the resource between the first processor and the second processor comprises arbitrating access to the resource between one or more applications running on the first processor (i.e. application programs; Figure 6-item 124) and one or more applications (i.e. legacy programs; Figure 8-item 224) running on the second processor core, (paragraphs [0050]-[0067]).

In reference to claim 3, Chennubhotla shows the resource sharing system wherein the first application layer entity is a resource manager, (i.e. memory controller/DMA) and the second application layer entity is a peer resource manager (i.e. memory controller/DMA), (Figure 2-item 104; column 5, lines 11-36).

In reference to claims 4 and 81, Dutta shows the system further comprising an application layer state machine (i.e. program) running on at least one of the first and second processors adapted to define a state (i.e. status information) of the resource, (paragraph [0065]).

In reference to claim 5, Chennubhotla shows the system further comprising an interprocessor resource arbitration messaging protocol, (i.e. interprocessor communications interface and protocol; columns 2-3).

In reference to claims 6 and 82, Dutta shows the system further comprising: for each of a plurality of resources (i.e. legacy devices; Figure 2-items 17A-17C) to be shared, a respective first application layer entity on the first processor (i.e. application programs; Figure 6-item 124) and a respective corresponding second application layer entity on the second processor (i.e. legacy programs; paragraphs [0062]-[0066]; Figure 8-item 224), the respective first application layer entity and the respective second application layer entity together being adapted to arbitrate access to the resource between the first processor and the second processor, using the first interprocessor communications

protocol (i.e. common communications protocol; paragraph [0028]), the physical layer interconnection and the second intercommunications protocol (i.e. respective protocols X.Y,Z; paragraph [0029]) to provide a communication channel between the respective first application layer entity and the respective second application layer entity, (paragraphs [0023]-[0033]).

In reference to claim 7, Dutta shows the resource sharing system wherein arbitrating access to each resource (i.e. legacy devices; Figure 2-items 17A-17C) between the first processor and the second processor comprises arbitrating access to the resource between one or more applications running on the first processor (i.e. application programs; Figure 6-item 124) and one or more applications (i.e. legacy programs; Figure 8-item 224) running on the second processor core, (paragraphs [0050]-[0067]).

In reference to claim 11, Dutta shows the resource sharing system further comprising for each resource to be shared a respective resource specific interprocessor resource arbitration messaging protocol, (i.e. distinct protocols X, Y, Z; paragraph [0029]).

In reference to claim 12, Dutta shows the resource sharing system further comprising for each resource a respective application layer state machine (i.e. legacy programs; Figure 8-item 224) running on at least one of the first and second processors adapted to define a state (i.e. status information) of the resource.

In reference to claim 13, Dutta shows the system wherein: the first interprocessor communications protocol and the second interprocessor communications protocol are adapted to provide a respective resource-specific communications channel in respect of each resource, each resource-specific communications channel providing an interconnection between the application layer entities arbitrating use of the resource, (i.e. translation program provides a device –specific proprietary interface to the interface object/legacy device; Figure 3; paragraphs [0030]-[0033]).

In reference to claim 14, Dutta shows the system wherein: the first interprocessor communications protocol and the second interprocessor communications protocol are adapted to provide a respective resource-specific communications channel in respect of each resource; wherein at least one resource-specific communications channel provides an interconnection between the application layer entities arbitrating use of the resource; wherein at least one resource-specific communications channel maps directly to a processing algorithm called by the communications protocol, (i.e. translation program provides a device –specific proprietary interface to the interface object/legacy device; Figure 3; paragraphs [0030]-[0033]).

In reference to claim 15, Chennubhotla shows the system wherein for each resource-specific communications channel, the first interprocessor communications protocol and the second interprocessor communications protocol each have a respective receive queue and a respective transmit queue, (i.e. each interprocessor communications interface each has a respective Receive FIFO buffer and Transmit FIFO Buffer; Figure 1-items 20-1 to 20-n; Figure 2-item 124&126; columns 4-6).

In reference to claim 16, Chennubhotla shows the system wherein the first and second interprocessor communications protocols are adapted to exchange messages using a plurality of priorities, (i.e. pre-determined arbitration algorithms; column 5, line 41 to column 6, line 55).

In reference to claim 17, Chennubhotla shows the system wherein the first and second interprocessor communications protocols are adapted to exchange data using a plurality of priorities by providing a respective transmit channel queue and a respective receive channel queue for each priority, and by serving higher priority channel queues before lower priority queues, (i.e. pre-determined arbitration algorithms; columns 6-9).

In reference to claim 18, Dutta shows the system wherein at least one of the application layer entities is adapted to advise at least one respective third application layer entity of changes in the state of their respective resources, (paragraphs [0062]-[0066]).

In reference to claim 19, Dutta shows the system wherein each at least one respective third application layer entity is an application which have registered (i.e. paragraphs [0034]-[0037]) with one of the application layer entities to be advised of changes in the state of one or more particular resources, (paragraphs [0062]-[0067]).

In reference to claim 20, Dutta shows the system wherein each state machine (i.e. legacy program; Figure 8-item 224) maintains a state of the resource and identifies how incoming and outgoing messages of the associated resource specific messaging protocol affect the state of the state machine, (paragraphs [0028]-[0033]; paragraphs [0062]-[0067]).

In reference to claim 27, Dutta shows the system wherein a state machine is maintained on both processors for each resource, (paragraphs [0050]-[0066]; Figures 6&8).

In reference to claim 28, Dutta shows the system wherein the second interprocessor communications protocol further comprises a system observable having a system state machine and state controller, (paragraphs [0050]-[0066]; Figures 6&8).

In reference to claim 29, Dutta shows the system wherein messages in respect of all resources are routed through the system observable, thereby allowing conglomerate resource requests, (i.e. operate under one proprietary protocol; paragraph [0033]; paragraphs [0050]-[0066]; Figures 6&8).

In reference to claim 49, Chennubhotla shows the system wherein the physical layer interconnection is a serial link, (Figure 1-item 18).

In reference to claim 50, Chennubhotla shows the system wherein the physical layer interconnection is an HPI (host processor interface), (Figure 1-item 20).

In reference to claim 51, Chennubhotla shows the system wherein the physical layer interconnection is a shared memory arrangement, (Figure 1-item 14; column 5).

In reference to claim 52, Chennubhotla shows the system wherein the physical layer interconnection comprises an in-band messaging channel and an out-of-band messaging channel.

In reference to claim 53, Chennubhotla shows the system wherein the out-of-band messaging channel comprises at least one hardware mailbox, (column 5; column 5).

In reference to claim 54, Chennubhotla shows the system wherein the at least one hardware mailbox comprises at least one mailbox for each direction of communication, (column 5; Figure 2).

In reference to claim 55, Chennubhotla shows the system wherein the in-band messaging channel comprises a hardware FIFO, (Figure 2-items 124&126).

In reference to claim 56, Chennubhotla shows the system wherein the in-band messaging channel comprises a pair of unidirectional hardware FIFOs, (Figure 2-items 124&126).

In reference to claim 57, Chennubhotla shows the system wherein the in-band messaging channel comprises a shared memory location, (Figure 2-item 14).

In reference to claim 58, Chennubhotla shows the system wherein the out-of-band messaging channel comprises a hardware mailbox, the hardware mailbox causing an interrupt on the appropriate processor, (columns 5-6).

In reference to claim 59, Chennubhotla shows the system wherein an out-of-band message to a particular processor causes an interrupt on the processor to receive the out-of-band message and causes activation of an interrupt service routine which is adapted to parse the message, (columns 5-6).

In reference to claim 76, Chennubhotla shows the system wherein the physical layer interconnection between the first processor and the second processor comprises an interprocessor interface (Figures 1&2-item 2; columns 2-3) for interfacing between a first processor core (i.e. processing subsystem; Figure 1-item 12-1; columns 4-5) and a second processor core (i.e. processing subsystem; Figure 1-item 12-2), the interprocessor interface comprising: at least one data FIFO queue (i.e. receive FIFO buffer; Figure 2-item 126) having an input adapted to receive data from the second processor core and an output adapted to send data to the first processor core; at least one data FIFO queue (i.e. transmit FIFO buffer; Figure 2-item 124) having an input adapted to receive data from the first processor core and an output adapted to send data to the second processor core; a first out-of-band message transfer channel (i.e. interprocessor communications bus; Figures 1&2-item 18) for sending a message from the first processor core to the second processor core; a second out-of-band message transfer channel (i.e. local bus; Figure 2-item 108) for sending a message from the second processor core to the first processor core, (columns 4-6).

In reference to claim 77, Chennubhotla shows the system wherein the interprocessor interface further comprising: a first interrupt channel adapted to allow the first processor core to interrupt the second processor core; and a second interrupt channel adapted to allow the second processor core to interrupt the first processor core, (columns 6-10; Figure 2-item 128).

In reference to claim 78, Chennubhotla shows the system wherein the interprocessor interface further comprising at least one register adapted to store an interrupt vector, (Figure 2-item 122; columns 6-10).

In reference to claim 79, Chennubhotla shows the system wherein the interprocessor interface having functionality accessible by the first processor core memory mapped to a first memory space understood by the first processor core, and having functionality accessible by the second processor core memory mapped to a second memory space understood by the second processor core, (columns 5-6; Figure 2-item 20).

Claims 8-10, 21-26, 30-48, and 83-86 are rejected under 35 U.S.C. 103(a) as being unpatentable over Chennubhotla and Dutta, as applied to claims 1 and 80, and further in vie w of Walter C. Dietrich, Jr. (Saving a Legacy with Objects [ACM]), hereinafter referred to as Dietrich.

In reference to claims 8 and 83, the Chennubhotla shows the system wherein the interprocessor communications protocol is designed to leave undisturbed real-time profiles of existing real-time functions of the processor running the other of the two interprocessor communications protocols, (columns 6-10). However the references, Chennubhotla and Dutta, fail to expressly show the system wherein one of the two interprocessor communications protocols is designed for efficiency and orthogonality between application layer entities running on the processor running the one of the two interprocessor communications protocols. Nonetheless, this was a well-known feature in the art at the time of invention, as further evidenced by Dietrich. Therefore, it would have been obvious to accordingly modify the system as disclosed by Chennubhotla and Dutta, for one of ordinary skill in the art at the time of invention.

In an analogous art, Dietrich discloses a legacy-based object-oriented interface to a legacy system, *(Introduction*, page 77). Dietrich further discloses that the system is designed for efficiency and orthogonality between application layer entities running on the processor, (*Increased Orthogonality*, page 81). One of ordinary skill in the art would have been so motivated to accordingly modify the system so as to increase system efficiency, (Dietrich; *Increased Orthogonality*, page 81).

In reference to claim 9, Chennubhotla shows the system wherein the first processor is a host processor, and the second processor is a coprocessor adding further functionality to the host processor, (column 1, lines 42-57; column 4, lines 44-65).

In reference to claim 10, Dutta shows the resource sharing system wherein the host processor has a message passing mechanism (i.e. operating system; paragraphs [0050]-[0054]; Figure 6-items 120) outside of the first interprocessor communications protocol to communicate between the first interprocessor communications protocol and the first application layer entity.
In reference to claim 21, Chennubhotla shows the system wherein the second interprocessor communications protocol comprises a channel thread domain which provides at least two different priorities over the physical layer interconnection, (i.e. bus arbiter; Figure 1-item 22; columns 8-9).

In reference to claim 22, Chennubhotla shows the system wherein the channel thread domain runs as part of a physical layer ISR (interrupt service routine), (columns 8-10).

In reference to claim 23, Chennubhotla shows the system wherein the channel thread domain provides at least two different priorities and a control priority, (i.e. pre-determined arbitration algorithms; columns 6-9).

In reference to claim 24, Dutta shows the system wherein for each resource, the respective second application layer entity comprises an incoming message listener, an outgoing message producer and a state controller, (paragraphs [0062]-[0066]; Figure 8).

In reference to claim 25, Dutta shows the system wherein the state controller and outgoing message producer are on one thread specific to each resource, and the incoming message listener is a separate thread that is adapted to serve a plurality of resources, (paragraphs [0062]-[0066]; Figure 8).

In reference to claim 26, Dutta shows the system wherein for each resource, the second application layer entity is entirely event driven and controlled by an incoming message listener, (paragraphs [0062]-[0066]; Figure 8).

In reference to claim 30, Dutta shows the system wherein each second application layer entity has a common API (application interface), (paragraphs [0062]-[0066]; Figure 8).

In reference to claim 31, Dutta shows the system wherein the common API comprises, for a given application layer entity, one or more interfaces in the following group: an interface for an application to register with the application layer entity to receive event notifications generated by this application layer entity; an interface for an application to de-register from the application layer entity to no longer receive event notifications generated by this application layer entity; an interface for an application to temporarily suspend the notifications from the application layer entity; an interface for an application to end the suspension of the notifications from that application layer entity; an interface to send data to the corresponding application layer entity; and an interface to invoke a callback function from the application layer entity to another application, (paragraphs [0034]-[0066]).

In reference to claim 32, Chennubhotla shows the system further comprising: for each resource a respective receive session queue and a respective transmit session queue in at least one of the first interprocessor communications protocol and the second interprocessor communications protocol, (i.e. respective transmit FIFO and Receive FIFO for each interprocessor communications interface; columns 5-9; Figures 1&2).

In reference to claim 33, Chennubhotla shows the system further comprising: for each of a plurality of different priorities, a respective receive channel queue and a respective transmit channel queue in at least one of the first interprocessor communications protocol and the second interprocessor communications protocol, (i.e. respective transmit FIFO and Receive FIFO for each interprocessor communications interface; columns 5-9; Figures 1&2).

In reference to claim 34, Chennubhotla shows the system further comprising on at least one of the two processors, a physical layer service routine adapted to service the transmit channel queues by dequeueing channel data elements from the transmit channel queues starting with a highest priority transmit channel queue and transmitting the channel data elements thus dequeued over the physical layer interconnection, and to service the receive channel queues by dequeueing channel data elements from the physical layer interconnection and enqueueing them on a receive channel queue having a priority matching that of the dequeued channel data element, (columns 6-10; Figure 2-items 124,122,126).

In reference to claim 35, Chennubhotla shows the system wherein on one of the two processors, the transmit channel queues and receive channel queues are serviced on a scheduled basis, the system further comprising on the one of the two processors, a transmit buffer between the transmit channel queues and the physical layer interconnection and a receive buffer between the receive physical layer interconnection and the receive channel queues, wherein the output of the transmit channel queues is copied to the transmit buffer which is then periodically serviced by copying to the physical layer interconnection, and wherein received data from the physical layer interconnection is emptied into the receive buffer which is then serviced when the channel controller is scheduled, (columns 6-10; Figure 2-items 124,122,126).

In reference to claim 36, Chennubhotla shows the system wherein each transmit session queue is bound to one of the transmit channel queues, each receive session queue is bound to one of the receive channel queues and each session queue is given a priority matching the channel queue to which the session queue is bound, the system further comprising: a session thread domain adapted to dequeue from the transmit session queues working from highest priority session queue to lowest priority session queue and to enqueue on the transmit channel queue to which the transmit session queue is bound, and to dequeue from the receive channel queues working from the highest priority channel

queue to the lowest priority channel queue and to enqueue on an appropriate receive session queue, the appropriate receive session queue being determined by matching an identifier in that which is to be enqueued to a corresponding session queue identifier, (columns 6-10; Figure 2-items 124,122,126).

In reference to claim 37, Chennubhotla shows the system wherein data/messages is transmitted between corresponding application layer entities managing a given resource in frames; wherein the session thread domain converts each frame into one or more packets; wherein the channel thread domain converts each packet into one or more blocks for transmission, (columns 5-7).

In reference to claim 38, Chennubhotla shows the system wherein blocks received by the channel controller are stored a data structure comprising one or more blocks, and a reference to the data structure is queued for the session layer thread domain to process, (columns 5-8).

In reference to claim 39, Chennubhotla shows the system further comprising, for each of a plurality of [queue, peer queue] pairs implemented by the first and second interprocessor communications protocols, a respective flow control protocol, (columns 6-9).

In reference to claim 40, Chennubhotla shows the system further comprising: for each of a plurality of [transmit session queue, peer receive session queue] pairs implemented by the first and second interprocessor communications protocols, a respective flow control protocol, wherein the session thread is adapted to handle congestion in a session queue; for each of a plurality of [transmit channel queue, peer receive channel queue] pairs implemented by the first and second interprocessor communications protocols, a respective flow control protocol, wherein the channel controller handles congestion on a channel queue, (columns 6-12).

In reference to claim 41, Chennubhotla shows the system wherein the session controller handles congestion in a receive session queue with flow control messaging exchanged through an in-band control channel, (columns 5-9).

In reference to claim 42, Chennubhotla shows the system wherein the physical layer ISR handles congestion in a receive channel queue with flow control messaging exchanged through an out-of-band channel, (columns 5-9).

In reference to claim 43, Chennubhotla shows the system wherein congestion in a transmit session queue is handled by the corresponding application entity, (columns 5-9).

In reference to claim 44, Chennubhotla shows the system wherein congestion in a transmit channel queue is handled by the session thread by holding any channel data element directed to the congested queues and letting traffic queue up in the session Queues, (columns 5-10).

In reference to claim 45, Chennubhotla shows the system wherein the interprocessor communications protocol designed to mitigate the effects on the real-time profile further comprises an additional buffer between the physical layer interconnection a scheduled combined channel controller/session manager function adapted to perform buffering during periods between scheduling of the combined channel controller/session manager, (columns 5-10).

In reference to claim 46, Chennubhotla shows the system wherein the first interprocessor communications protocol interfaces with application layer entities using a message-passing mechanism provided by the processor the external to the first interprocessor communications protocol of the first processor, each application layer entity being a resource manager, (columns 5-10).

In reference to claim 47, Chennubhotla shows the system wherein the first interprocessor communications protocol is implemented with a single thread acting as a combined channel controller and session manager, (columns 5-10).

In reference to claim 48, Chennubhotla shows the system wherein the first interprocessor communications protocol is implemented with a single system task acting as a combined channel controller and session manager, (columns 5-10).

In reference to claim 84, Dutta shows the resource sharing system further comprising for each resource to be shared a respective resource specific interprocessor resource arbitration messaging protocol, (i.e. distinct protocols X, Y, Z; paragraph [0029]); and for each resource a respective application layer state machine (i.e. legacy programs; Figure 8-item 224) running on at least one of the first and second processors adapted to define a state (i.e. status information) of the resource.

In reference to claim 85, Chennubhotla shows the system wherein the first and second interprocessor communications protocols are adapted to exchange data using a plurality of priorities by providing a respective transmit channel queue and a respective receive channel queue for each priority, and by serving higher priority channel queues before lower priority queues, (i.e. pre-determined arbitration algorithms; columns 6-9).

In reference to claim 86, Dutta shows the system wherein each state machine (i.e. legacy program; Figure 8-item 224) maintains a state of the resource and identifies how incoming and outgoing messages of the associated resource specific messaging protocol affect the state of the state machine, (paragraphs [0028]-[0033]; paragraphs [0062]-[0067]).

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to LaShanya R Nash whose telephone number is (571) 272-3957. The examiner can normally be reached on 9am-5pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Glenton Burgess can be reached on (571) 272-3949. The fax phone number for the organization where this application or proceeding is assigned is (571) 273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

LaShanya Nash
Art Unit, 2153
August 18, 2005

GLENTON B. BURGESS
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100

Notice of References Cited

Application/Control No.	Applicant(s)/Patent Under Reexamination	
09/941,619	COMEAU ET AL.	
Examiner	Art Unit	Page 1 of 1
LaShanya R. Nash	2153	

U.S. PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Name	Classification
	A	US-4,901,231	02-1990	Bishop et al.	707/205
	B	US-5,841,988	11-1998	Chennubhotla et al.	709/237
	C	US-4,387,427	06-1983	Cox et al.	718/102
	D	US-2001/0047383	11-2001	Dutta, Prabal K.	709/201
	E	US-2002/0116454	08-2002	Dyla et al.	709/203
	F	US-5,682,534	10-1997	Kapoor et al.	719/328
	G	US-			
	H	US-			
	I	US-			
	J	US-			
	K	US-			
	L	US-			
	M	US-			

FOREIGN PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Country	Name	Classification
	N					
	O					
	P					
	Q					
	R					
	S					
	T					

NON-PATENT DOCUMENTS

*		Include as applicable: Author, Title Date, Publisher, Edition or Volume, Pertinent Pages)
	U	Dietrich, W.C.; Nackman, L.R.; Gracer, F.,"Saving legacy with objects",ACM SIGPLAN Notices, Volume 24 Issue 10, September 1989, pp.77-83; [retrieved on 16-08-05 ACM Database].
	V	
	W	
	X	

*A copy of this reference is not being furnished with this Office action. (See MPEP § 707.05(a).)
Dates in MM-YYYY format are publication dates. Classifications may be US or foreign.

Saving a Legacy with Objects

WALTER C. DIETRICH, JR.
LEE R. NACKMAN
FRANKLIN GRACER

Manufacturing Research Department
IBM Research Division, Thomas J. Watson Research Center
Yorktown Heights, NY 10598

Abstract: Developers of application software must often work with "legacy systems." These are systems that have evolved over many years and are considered irreplaceable, either because it is thought that duplicating their function would be too expensive, or because they are trusted by users. Because of their age, such systems are likely to have been implemented in a conventional language with limited use of data abstraction or encapsulation. The lack of abstraction complicates adding new applications to such systems and the lack of encapsulation impedes modifying the system because applications depend on system internals. We describe our experience providing and using an object-oriented interface to a legacy system.

1. INTRODUCTION

Developers of application software systems must often work with "legacy systems." These are systems that have evolved over many years and are considered irreplaceable, either because re-implementing their function is considered to be too expensive, or because they are trusted by users. Because of their age, such systems are likely to have been implemented in a conventional procedural language with limited use of data abstraction or encapsulation. The lack of abstraction complicates adding new applications to such a system and the lack of encapsulation impedes modifying the system itself because applications come to depend on system internals. We describe in this paper our experience in providing and using an object-oriented programmer's interface to a legacy system.

The legacy system used in our study is the Geometric Design Processor (GDP) [Wesley80] [Wolfe87], a solid modeling system consisting of several hundred thousand lines of PL/I code which has evolved over 15 years. Solid modeling systems provide means for representing, manipulating, and analyzing models of three-dimensional solid objects. They also provide facilities for managing large assemblies of parts, bills of material, dimensions and tolerances, etc. GDP is used routinely in a production environment for the mechanical design of IBM mainframe computers [Wolfe87].

In a production environment, user requests for new function must be satisfied quickly. This has led to the evolution of many PL/I procedures for creating and manipulating solids, managing data, and interacting with the user. These have been collected and organized into a programming interface. The system described here, the *Tiered Geometric Modeling System* (*TGMS*), provides an alternative, object-oriented interface to the GDP programming interface. We believe that such an interface provides a higher-productivity environment for developing new applications.

The overall architecture of *TGMS* is shown in Figure 1. It consists of GDP, interfaced to the interpreter for AML/X (an object-oriented programming language intended for use in design and manufacturing applications [Nackman86]), together with some AML/X class definitions. The class definitions constitute the programming environment seen by *TGMS* users.

This paper discusses important issues that arose in *TGMS's* design and implementation, emphasizing the issues that we believe will arise in providing an object-oriented interface to any legacy system. We conclude with a synopsis of *TGMS* user experiences.

2. DESCRIPTION OF THE TIERED GEOMETRIC MODELING SYSTEM

2.1 Description of the Legacy System: GDP is an interactive graphic system for modeling three-dimensional objects, especially mechanical parts and assemblies. GDP provides several kinds of simple solids, called *primitives;* primitive types are cuboid (rectangular block), cylinder, cone, hemisphere, translated polygon and rotated polygon (solids of translation and revolution). Complex solids are created by combining solids using set operations (union, intersection, and difference).

Various geometric operations can be performed on solids, such as checking for alignment and interference. Several physical properties (e.g., mass) can also be calculated. A tree is used to represent the structure of the product being modeled. Solids are associated with nodes in the tree: leaf nodes correspond to primitives, internal nodes to combinations of primitives.

GDP is written in PL/1 and runs on IBM mainframe computers, using a variety of graphics workstations. It was written and evolved as a monolithic system, but recently, selected subroutines in GDP were modified and documented to form an application programming interface. The programming interface has several subsets. One subset performs functions required for user interaction, such as pointing and menu display. Another subset allows the programmer to create and manipulate solids.

2.2 Overview of *TGMS*: The classes in *TGMS* are designed to be easy to use but powerful. This is achieved by keeping several design goals in mind: orthogonality, conciseness, and compatibility with AML/X. *Orthogonality* means that a single method or subroutine doesn't perform logically separate functions. It usually makes systems easier to learn and use. By *compatibility with AML/X,* we mean that the semantics of the class should be similar to those of the intrinsic AML/X types. For example, `a=b` means assign a copy of `b` to `a` if `a` and `b` are both instances of the same built-in type; it should mean the same if they are both instances of the solid object class. *Conciseness* has the obvious meaning. When dealing with classes that represent mathematical entities, it is possible to design a concise notation that is easy to read by building on our experience with equations and formulas.

TGMS has classes for these low-level geometric entities: point, vector, and transformation matrix [Rossigna87]. A transformation matrix can be used to represent a translation, a rotation, a change in size (scaling), or a combination of these. The transformation matrix classes have several class methods that

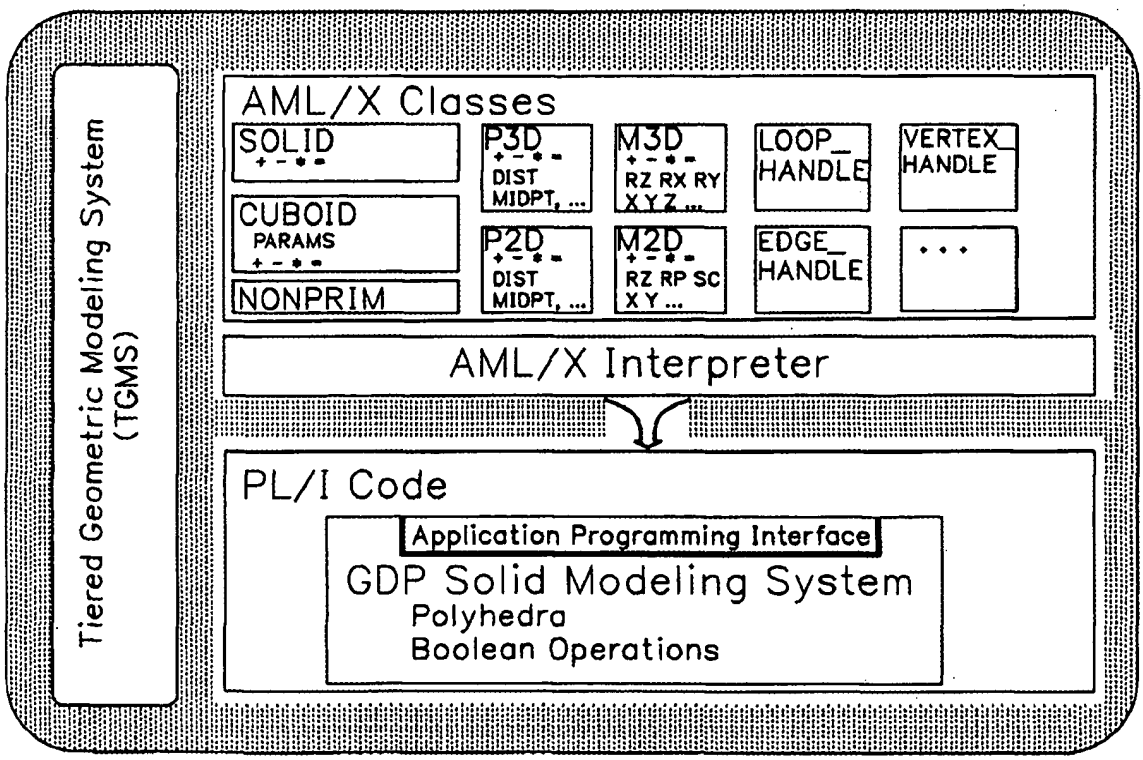

Figure 1. High-level structure of *TGMS*
This figure is based on a figure in [Dietrich89] copyright (1989). Reprinted by permission of John Wiley & Sons, Ltd.

create basic transformations such as translations and rotations about the axes. These classes overload the AML/X arithmetic operators so that concise and easily understood expressions can be written using these classes. For example, if `point1` is a point, `vec1` is a vector, and `tmat1` and `tmat2` are transformation matrices, `point1+vec1` evaluates to the point that results from adding `vec1` to `point1`, `point1*tmat1` yields the point that results from applying the transformation matrix `tmat1` to `point1`, and `tmat1*tmat2` evaluates to the transformation that would result from applying `tmat1` followed by `tmat2`.

The `solid` class is used to represent three-dimensional solid objects. As shown in Figure 2, the `solid` class has subclasses `primitive` and `nonprim`.

The `primitive` class has one subclass for each kind of primitive solid. Instances are created by sending messages to the appropriate classes, giving as arguments the parameters that define the shape, e.g., height and radius for cylinder, polygon and distance for extrusion.

The `nonprim` class has two subclasses: `hull`, for solids created by computing the convex hull of a set of points, and `boolean_combination`, for solids created by applying union, intersection, or difference (subtraction) to a pair of solids.

Instances of `solid` can be created by sending messages to the these subclasses. Instances can also be created by using overloaded arithmetic operators to move solids in space, change their size, or perform Boolean operations. For example, evaluating `(solid1+solid2)+vec1` results in a new object (of type `union`) that is the union of `solid1` and `solid2`, translated in the direction and the distance specified by the vector `vec1`.

Instances can be changed using the assignment operators (=, +=, etc.). The `solid` class overloads +=, -=, and *= so that operations that modify a solid can be written more concisely. Another advantage of this

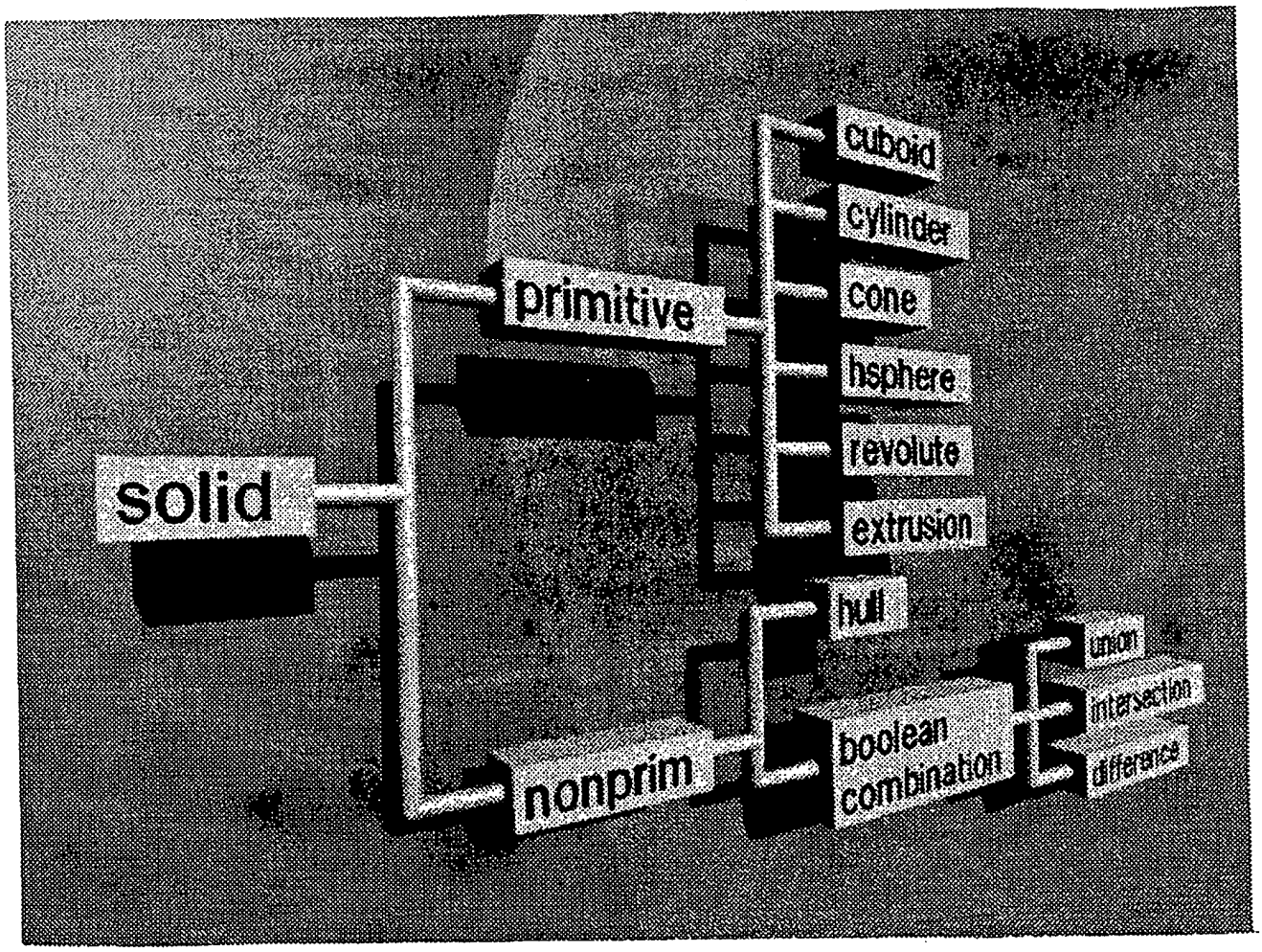

Figure 2. Hierarchical structure of the classes for solid objects
In this rendition, each class's name is closer to the viewer than its subclasses names. This ray-traced image was generated with *TGMS*, using one simulated light source. (The wall behind the tree contains the shadow of the tree and a reflection of its back.)

notation is that using an operator such as += can be more efficient if the modeler's algorithm for the operation can update in place.

The class hierarchy is a natural result of the properties of the different kinds of solids. The primitive subclasses have methods that return creation parameters of their instances, whereas other subclasses do not. The creation parameters of a primitive are the basic data values that, taken together with the primitive's type and its orientation, provide an exact and very concise description of the solid.

TGMS also contains classes that allow the boundaries (faces, edges, vertices, etc.) of solids to be queried. More complete descriptions of all of the classes, with sample programs, appear in [Dietrich88, Dietrich89].

3. DESIGN ISSUES

In designing an object-oriented system, the choice of objects and methods is crucial. The system builder is typically free to make design decisions that result in the most natural metaphor for the system's application domain. Unfortunately, since this is not true when legacy-based object-oriented systems are being developed, a host of new design issues are raised.

Two reasons for implementing an object-oriented system using a legacy system (*legacy*) are

- the legacy satisfies most of the needs of its users, but a better programming interface is needed for extending or customizing the functions of the legacy, or
- the legacy has code that can be reused to implement (part of) a desired object-oriented system's function.

We built *TGMS* for both of these reasons.

An important improvement provided by *TGMS* is to isolate its applications from either modification or replacement of the underlying solid modeler. The GDP programming interface defines data structures and procedures for representing and manipulating solids and lower-level geometric entities. By encapsulating these data structures, *TGMS* prevents the user from introducing knowledge of the GDP programming interface into new applications. Those GDP functions accessible through the object-oriented interface are also made to appear to be more complete and orthogonal, and thus easier to use.

The second reason for building *TGMS* resulted from a desire to reuse complex, trusted portions of the GDP code (especially the polyhedral set operations) to implement high-level geometric objects.

Building a single object-oriented system for both reasons requires design compromises. To provide a programming interface for extending GDP's function, the interface must include essentially all of GDP's function, not just its geometric operations. However, *TGMS* does not provide access to all of GDP, although most *TGMS* applications require access to some non-geometric parts of GDP, such as the filing system and the interactive user interface. For example, the GDP model is stored as a tree which is used heavily by GDP applications to aggregate solids and to attach attributes to them. This tree is not exposed by *TGMS* because more general data structures are simple to implement as classes. This design provides simplicity and generality, but it does not satisfy *TGMS* applications that need to access the tree (e.g. classes which use models created in GDP sessions.) Therefore, we have had to provide minimal access to GDP's tree despite the loss of simplicity. We believe that one of the most important and difficult aspects of designing a legacy-based object-oriented system is making the trade-off between exposing the legacy's functions and isolating object-oriented applications from changes in the legacy.

This decision is more difficult when the legacy provides an interactive interface as well as a programmer's interface. Should the object-oriented system's end-user interface be based on the same subroutines as the legacy interface? If so, it may be difficult for object-oriented applications to provide the end-user interfaces their users need. For example, GDP's interactive user interface is intended for use by mechanical designers; applications written in *TGMS* often need to present a very different user interface. Nonetheless, user interfaces for three-dimensional geometry applications usually require various rendering algorithms which are expensive to reimplement. For *TGMS*, we exposed GDP's interactive interface and also provided programmer access to the subroutines that control the interface (e.g. prompting, menu selection, and viewing and rendering). Thus far, this compromise has been satisfactory.

4. EXTERNAL OBJECT IMPLEMENTATION

TGMS was implemented by writing AML/X classes that call GDP subroutines and refer to its data structures. The code that encapsulates the legacy is called the *wrapper*. The use of the legacy code is called an *external object implementation* since we do not directly implement the objects in the object-oriented component. For external object implementations to be successful, several problems must be solved [Dietrich88].

Interlanguage communication: To build an external object implementation, one needs more than simply the ability to call subroutines in the legacy's language. Since the wrapper depends on data that is internal to the legacy, the inter-language communication mechanism must also preserve the legacy's state when legacy subroutines called from the wrapper return.

Garbage collection: If the legacy does its own garbage collection, it must be prevented from collecting data that is referenced by the wrapper. One way to do this is to ensure that for every pointer in the wrapper that refers to legacy data, there is a corresponding (non-garbage) pointer in the legacy that refers to the same data.

Memory compaction: If the wrapper has pointers to the legacy's data and the legacy does compaction, the legacy may move referenced data without updating the wrapper's pointers. There are two solutions. In the first, pointers in the wrapper refer indirectly to legacy data through a table of pointers in the legacy; it must be possible for the wrapper to access the table in a compaction-invariant way. This solution also solves the garbage-collection problem. An alternate solution is not to use pointers from the wrapper to legacy's data. This solution is used in *TGMS* because GDP has memory compaction but not garbage collection.

If pointers in the legacy can also refer to data in the object-oriented part of the legacy-based object-oriented system, the garbage collection and memory compaction problems can occur in both directions. The same techniques may be used to solve the problems in each direction, but it may be necessary to coordinate the two languages' garbage collectors. In *TGMS*, GDP never refers to the wrapper's data.

Object lifetime synchronization: Generally, if block exit or garbage collection deallocates an object in the wrapper, the corresponding legacy object should be freed or made available for garbage collection. This is only practical if wrapper's language has *destroy methods* [Atkins88, Dietrich89]. We therefore consider destroy methods to be essential for external object implementations.

Cross-system consistency: The preceding problems are well-defined, concrete, and relatively easy to solve. For external object implementations to be successful, one must also carefully control how the legacy and wrapper interact. *Cross-system invariants* are assertions that relate data in both the legacy and wrapper, for example, "For every instance of class *c* in the wrapper, there is exactly one data block (or structure) of type *d* in the legacy". Even if the wrapper's language does not support assertion checking, cross-system invariants are a useful tool during design and debugging.

Cross-system invariants may be violated if the memory management problems described above are not solved. They may also be violated if interrupts are not taken into consideration during design, as our experience illustrates: When a `solid` instance is deallocated, its destroy method deallocates the corresponding GDP polyhedron. In the AML/X environment, users can interrupt the execution of any method, abort execution of the program, and cause the interpreter to resume the read-eval-print loop. If this is done during execution of a solid's destroy method, the `solid` instance will be deallocated but the GDP polyhedron may not be, depending on when the destroy method was interrupted. Because of this general problem, we have concluded that the semantics of AML/X destroy methods should be changed [Atkins88]. This also applies to other languages that have both destroy methods and interrupt handling.

Since the legacy frequently has an attractive end-user interface, it is tempting to allow end-users of the legacy-based object-oriented system to use the legacy interface directly (perhaps in conjunction with one designed specifically for the legacy-based object-oriented system). This is especially true early in the life of a legacy-based object-oriented system, when its function may be more important than its user interface. Unfortunately, it is often possible to violate cross-system invariants through direct use of the legacy's end-user interface. This suggests that legacy-based object-oriented system's should only expose a restricted version of the legacy user interface which only allows data to be queried and displayed, not modified.

Increasing Orthogonality: A design goal for any legacy-based object-oriented system should be to present orthogonal classes. Lack of orthogonality in the legacy makes this difficult to achieve because the wrapper implementers must understand all of the undesired interactions in the legacy. *TGMS* provides a good example of this. The stand-alone version of GDP always keeps the displayed view of the model up-to-date with respect to the actual model, which relieves the user from having to account for differences between the displayed model and the actual model. Unfortunately, some of the modeling subroutines in the first version of the GDP programming interface also updated the display.

Synchronizing the display with the model violates orthogonality because graphics and modeling are different functions. As a practical matter, synchronizing them would make *TGMS* less useful for new solid modeling applications because new high-level operations that require several solid modeling steps would involuntarily update the display several times. This would distract end users (and be inefficient). We could have hidden this non-orthogonality but it would have required us to either know which GDP subroutines caused the display to be updated and under what conditions, or to encapsulate the entire GDP user interface. The first alternative was too complicated and the second was beyond the scope of the project. Partly as a result of our experience, the second version of the GDP programming interface was designed with a clean split between modeling and graphics. This increase in orthogonality was evident in other areas of the programming interface as well, e.g. modeling and input. We believe that building a legacy-based object-oriented system on a legacy that lacks orthogonality requires participation of someone with expertise in the architecture and internals of the legacy.

5. CONCLUSIONS

TGMS came into use at the IBM Thomas J. Watson Research Center in 1986. It has not been tested and documented enough to be a production system, but it has been used to build five different applications. The largest application is WADE, a Workcell Application Development Environment [Levas89]. WADE helps users design and test robotic workcells, and contains classes for equipment found in workcells (e.g., robots, sensors, computers, conveyer belts, feeders) and for concepts related to workcells such as trajectories, events, and programs. Users lay out workcells by creating class instances and positioning them or associating them with other class instances. For example, `robot` instances are created and positioned and `program` instances are created and associated with the robots. Once all of the pieces of equipment are set up and associated with a `workcell` instance, the user can tell the workcell to simulate its execution. Using the kinematics and dynamics of the equipment in the workcell, WADE gives a temporally accurate simulation of the workcell. Because it uses the `solid` class to represent the geometric state of the workcell, visualization, static interference analysis (to make sure parts do or don't touch), and other geometric analyses are easy. WADE contains about 50 classes with about 670 methods.

Two new applications of *TGMS* are in the areas of machine vision and automated machining. Given a solid and a set of features (edges, faces, and vertices), the vision application [Tarabani89, Tsai89] computes the places where a visual sensor can be positioned to view all of the features without occlusion by the solid. It makes heavy use of the boundary representation classes and the Boolean operations. It is written procedurally and contains about 110 subroutines. The other new application is designed to decide what kind of numerically controlled milling machines are able to create a part, given its solid model. This is the first step in the automated creation of parts from their models. The program, based on an algorithm described in [Srinivas88], recognizes machining process features using the boundary representation and primitive creation parameters of the model's primitives. It uses an attributed finite state grammar to describe ideal machining process features. The actual features present on a part, which may be different from the ideal features because of feature interaction, are recognized using the grammar. The preliminary implementation is almost complete and contains about 60 subroutines.

TGMS users have found it to be easy to learn, easy to use, and powerful. In the words of Levas and Jayaraman [Levas89],

> The use of a production quality geometric modeling system in [WADE] contributed significantly to the capabilities of the resulting environment. The modeling of the geometric components of the equipment and simulation of many processes such as material removal and sensor based motions are directly attributable to the functionality provided by the underlying modeling system (GDP). The ease of interaction with the modeling system that *TGMS* provided facilitated the system development and user interaction.

We believe that this results from achieving the goals of orthogonality, expressiveness, conciseness, extensibility, and simplicity. Conciseness derives from liberal use of operator overloading and powerful aggregation and mapping facilities in AML/X. Expressiveness derives from conciseness and from design decisions that made the solid classes consistent with the language.

6. FUTURE WORK

The preceding discussion applies to situations in which one legacy is used as part of an object-oriented system. It should be clear that we are not restricted to one legacy per object-oriented system. Several legacy systems could be integrated to build a very powerful object-oriented environment. An example can be taken from the computer-aided design field. A system for computer design should do many things, including three-dimensional design of components (frames, cables, etc.), analysis of their properties (e.g. strength and resistance), and inspection of electrical connections. We might do this in one object-oriented system using one legacy for solid modeling, another for analysis (a partial differential equation solver) and a third legacy for verifying end-to-end connection of wires.

7. ACKNOWLEDGEMENTS

The idea of building object-oriented systems on top of legacy systems, and of using this as an integrating concept as described in the previous section, arose during discussions among Mark Lavin, Lee Nackman, and Michael Wesley on architectures for computer-assisted design and manufacturing systems. Christine Sundaresan helped design and implement *TGMS*. Jarek Rossignac contributed the low-level geometry classes [Rossigna87] and was an early user of the system. Anthony Levas and Rangarajan Jayaraman decided to build WADE on *TGMS* before implementation of *TGMS* had even begun. Seeing their system running provided inspiration and helped convince us that we were on the right track. Kostantinos Tarabanis, Roger Tsai, Stephen Byers, and Ramesh

Srinivasan, our newest users, helped show us where *TGMS* could be improved and sometimes made the improvements for us.

We thank Michael Karasick and Jarek Rossignac for providing many helpful comments on an early version of this paper. Special thanks go to John Barton for his many insightful comments on this paper.

REFERENCES

Atkins88. Atkins, M., and Nackman, L. The Active Deallocation of Objects in Object-Oriented Systems. *Software—Practice and Experience*, 18(11):1073-1089, November 1988.

Dietrich88. Dietrich Jr., W. C., Nackman, L. R., Sundaresan, C. J., and Gracer, F. TGMS: An Object-Oriented System for Programming Geometry, IBM Thomas J. Watson Research Center, RC 13444, Yorktown Heights, NY. January 1988.

Dietrich89. Dietrich Jr., W. C., Nackman, L. R., Sundaresan, C. J., and Gracer, F. TGMS: An Object-Oriented System for Programming Geometry. *Software—Practice and Experience*, 19(10), October 1989.

Levas89. Levas, A., and Jayaraman, R. WADE: An Object-Oriented Environment for Modeling and Simulation of Workcell Applications. *IEEE Transactions on Robotics and Automation*, 5(3):324-336, June 1989.

Nackman86. Nackman, L. R., Lavin, M. A., Taylor, R. H., Dietrich Jr., W. C., and Grossman, D. D. AML/X: A Programming Language for Design and Manufacturing. *Proc. Fall Joint Computer Conference*, pages 145-159, November 1986.

Rossigna87. Rossignac, J. R. AML/X tools for primitive geometric calculations: Points, Vectors, Coordinate Frames, and Linear Transformations, IBM Thomas J. Watson Research Center, RA 189, Yorktown Heights, NY. May 1987.

Srinivas88. Srinivasan, R., and Ferreira, P. Geometric Models of Machining Processes for Computer-Aided Process Planning. *Geometric Modeling for Product Engineering, selected papers from the IFIP/NSF Workshop on Geometric Modeling*, Rensselaerville, NY, September 1988. To appear.

Tarabani89. Tarabanis, K., and Tsai, R. Y. Viewpoint Planning: The Visibility Constraint. *Proc. of DARPA Image Understanding Workshop*, Palo Alto, California, May 27 1989. To appear.

Tsai89. Tsai, R. Y., and Tarabanis, K. Sensor Placement Planning. *Proc. Third Machine Vision Workshop*, New Brunswick, New Jersey, April 1989. To appear.

Wesley80. Wesley, M. A., Lozano-Perez, T., Lieberman, L. I., Lavin, M. A., and Grossman, D. D. A Geometric Modeling System for Automated Mechanical Assembly. *IBM Journal of Research and Development*, 24(1):64-74, January 1980.

Wolfe87. Wolfe, R., Wesley, M., Kyle Jr., J., Gracer, F., and Fitzgerald, W. Solid modelling for production design. *IBM Journal of Research and Development*, 31(3):277-295, May 1987.

This Page is Inserted by IFW Indexing and Scanning Operations and is not part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☑ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☑ FADED TEXT OR DRAWING
- ☐ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☑ GRAY SCALE DOCUMENTS
- ☑ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: _____

IMAGES ARE BEST AVAILABLE COPY.
As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.